```python
from sklearn.datasets import load_breast_cancer
import numpy as np
import random
import matplotlib.pyplot as plt
(data, target) = load_breast_cancer(return_X_y=True)

"""
1. Your K-means function should take in:
- an array containing a dataset
- a value of K,
and return:
- the cluster centroids
- the cluster assignment for each data point.
"""
def distortion(data, centroids, cluster_assignment):
    distortionValue = 0
    for i in range(len(data)):
        centroid = centroids[cluster_assignment[i]]
        distortionValue += np.linalg.norm(data[i] - centroid) ** 2
    distortionValue /= len(data)
    return distortionValue

def distance(data, centroid):
    return np.linalg.norm(data - centroid)

def kmeans(data, K):

    # Initialize centroids randomly
    centroids = []
    positions = []
    for i in range(K):
        position = random.randint(0, len(data)-1)
        while position in positions:
            position = random.randint(0, len(data))
        positions.append(position)
        centroids.append(data[position])

    # assign each data point to the closest centroid

    cluster_assignment = []
    for i in range(len(data)):
        distanceToCentroids = []
        for centroid in centroids:
            distanceToCentroids.append(distance(data[i], centroid))
        cluster_assignment.append(np.argmin(distanceToCentroids))

    # recompute centroids based on the assignments
    newCentroids = []
    for i in range(K):
        cluster = []
        for j in range(len(data)):
            if cluster_assignment[j] == i:
                cluster.append(data[j])
        newCentroids.append( np.mean(cluster, axis=0))

    # repeat until convergence
    while np.array_equal(centroids, newCentroids) == False:
        centroids = newCentroids

        # assign each data point to the closest centroid
        for i in range(len(data)):
            distanceToCentroids = []
            for centroid in centroids:
                distanceToCentroids.append(distance(data[i], centroid))
            cluster_assignment.append(np.argmin(distanceToCentroids))

        # recompute centroids based on the assignments
        for i in range(K):
            cluster = []
            for j in range(len(data)):
                if cluster_assignment[j] == i:
                    cluster.append(data[j])
            newCentroids[i] = np.mean(cluster, axis=0)

    return centroids, cluster_assignment
```

```
"""
2. Run the K-means algorithm for values of K varying between 2 and 7, at increments of 1.
Justify in your answer which data you passed as the input to the K-means algorithm.

3. Plot the distortion achieved by K-means for values of K varying between 2
and 7, at increments of 1.
"""
distortions = []
for i in range(2, 8):
    distortionValue = 0
    for j in range(100):
        centroids, cluster_assignment = kmeans(data, i)
        distortionValue += distortion(data, centroids, cluster_assignment)
    distortionValue /= 100
    print("K = " + str(i) + " Distortion = " + str(distortionValue))
    distortions.append(distortionValue)

plt.plot(range(2, 8), distortions)
plt.xlabel("K")
plt.ylabel("Distortion")
plt.title("Distortion vs K")
plt.legend(['Average Distortion of 100 runs'])
plt.show()

"""
4. (1 point) If you had to pick one value of K, which value would you pick? Justify your
choice.
"""
```